Measuring Software Requirements Specification Quality

Azlin Nordin, Nurul Husna Ahmad Zaidi and Noor Asheera Mazlan
International Islamic University Malaysia, Gombak, Kuala Lumpur, Malaysia.
azlinnordin@iium.edu.my

***Abstract*—The quality of a Software Requirements Specification (SRS) is measured in terms of quality properties such as completeness, conciseness, consistency and understandability. In general, evaluation of the SRS quality is done manually during review sessions. The evaluation process, however, is hugely dependent on the expertise of human experts i.e. the reviewers. In fact, the judgment of the human experts could also be inconsistent due to various factors including experience, knowledge and domain. The objectives of this study are to (1) identify feasible rules to measure SRS quality; and (2) help requirements engineer to improve their SRS quality. In this study, we analyzed SRS quality properties from the literature and identified quality factors that are feasible to be automated. From here, we identified two types of properties that are (1) requirements sentence quality (RSQ) and (2) requirements document quality (RDQ). For each of the type, its relevant quality indicators were identified. From here, rules on how to identify the quality indicators were further investigated and documented. As a case study, we implemented *SRS Quality-Checker* tool as a *proof-of-concept* for demonstrating how the rules were implemented to measure the SRS quality.**

***Index Terms*—Measuring SRS Quality; Requirements Document Quality; Requirements Review; Requirements Sentence Quality.**

I. INTRODUCTION

Requirements are features of the system-to-be-built that are discovered and identified before building any products or system. It is a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents [1]. Requirement Engineering (RE) is a systematic process to gather requirements from different sources and implement them into the software development life cycle [2]. RE is generally performed during the early stages of software development lifecycle. RE contains a set of activities that are discovering, analyzing, documenting, validating and maintaining a set of requirements for a system [3]. The validation activity involves evaluation and verification of requirements such as review, testing, and inspection. The final outcome of the RE process is the Software Requirement Specification (SRS), which contains the needs of the stakeholders and its constraints.

A clear set of requirement statements is one of the critical success factors to ensure project success [4]. While opinions on why projects are impaired and ultimately cancelled ranked incomplete requirements as the top of the list [4]. As can be observed, requirements give a significant impact to success of software projects. Hence, it is important to ensure that quality of SRS is accomplished to support the achievement of any software project.

An SRS is a document that describes all the externally observable behaviors and characteristics expected of a software system [5]. It is important to the developers as it allows them to save time on communication, minimize development efforts, gives the customer feedback, eliminates task duplication, facilitates the transfer to new users or to new machines, and breaks problems into parts. Furthermore, it also serves as the main document to verify validation and testing processes. There is no standard way to write an SRS document. Nonetheless, a good SRS should contain all the information as suggested in the IEEE Recommended Practice for SRS [6].

A poor requirement cannot lead to excellent software because the quality of any product depends on the quality of SRS itself [4]. Moreover, not all software developers are being trained to properly document and verify the quality of requirements in the SRS.

Quality Properties of SRS is one that contributes to successful, cost-effective creation of software that solves real user needs [5] and some of the qualities are:

i. Conciseness
   An SRS is a concise if, and only if, every requirement stated therein has only one interpretation [6]. This criterion possibly is the most difficult attribute to achieve using natural language. Though, the third-party inspector can determine the existence of conciseness by answering the predefined closed questions.
ii. Understandability
   The capability of each requirement to be fully understood by the stakeholders when they used it for developing software.
iii. Completeness
   An SRS is complete if, and only if, it includes the following elements [6]:
   a) All significant requirements, whether involving functionality, performance, design constraints, attributes, or external interfaces.
   b) Complete labels and references to all figures, tables, and diagrams in the SRS and definition of all terms and units of measure.

During requirement validation activity, requirement review is conducted to measure the SRS quality. Requirement review involves several independent inspectors that individually analyze the SRS to search for defects, and then meet to discuss the findings and recommend appropriate actions for fixing the agreed defects [7]. The purpose of conducting the requirement review is to find errors and point out other matters of concern in the requirement specification.

One of the challenges of requirement review is when there is a lack of ability to recognize the defect in the requirement

[8]. It will be tedious for the reviewers to search the errors in the requirements due to factors including time constraint and different background of knowledge and understanding. Despite that, the real problem arises is inconsistent with judgment among the human experts [9]. The reason for that is due to the different background of expertise and knowledge.

Hence, we suggested a solution to reduce the challenges and issues in such problems by analyzing and proposing the Quality Properties that are feasible to be implemented in our *SRS Quality Checker* tool. The purpose of this paper is to identify the Quality Properties rules to measure the SRS quality. Hence, this can help the requirement engineers to improve their SRS quality.

This paper is divided into five sections. The Introduction section lays out the background details, while the rest of the paper is organized as follows: Section II provides the literature review that briefly summarizes the aspects of SRS quality, while Section III presents the Research Methodology that we applied in this study. Next, Section IV provides the discussion regarding our study and finally, Section V concludes this paper.

II. Literature Review

In this paper, we analyzed two research papers that discussed the Quality Attributes to determine the quality of SRS. The papers are:

i. Quality Evaluation of Software Requirements Specification [10];
ii. Writing Effective Requirements Specification [11].

After we analyzed both the papers, we scoped this research paper to two aspects of the SRS quality [9]:

i. Requirement Sentence Quality (RSQ): the syntactical quality of single sentences considered separately;
ii. Requirement Document Quality (RDQ): the quality of the sentences considered in the context of the whole requirements documents.

From the papers, we finalized the quality attributes that are feasible to be automated by using the rules to determine the RSQ and RDQ that will further be discussed in the subsequent subsection.

A. Quality Attributes

The definition of each of the Quality Properties has been discussed in the Introduction, while this section will briefly explain the definition of each of the Quality Attributes. The Quality Attributes are divided into attributes related to RSQ and RDQ. The following are Quality Attributes with its brief definition of each attribute.

1) RSQ related attributes:

a) Implicit Sentences

A sentence is an implicit subject sentence if:

- its subject contains a demonstrative adjective;
- is expressed by means of pronouns;
- is specified by prepositions and is specifies by an adjective [10].

b) Optional Sentences

A sentence is optional if it contains an option [10].

c) Vague Sentences

A sentence is vague if it includes words holding inherent vagueness [10].

d) Weak Sentences

Category of clauses that will cause uncertainty and leave room for multiple interpretations [11].

e) Multiple Sentences

A sentence is multiple if it has more than one subject or more than one main verb [10].

f) Directives

Category of words and phrases that point to illustrative information within the requirement document. The data and information pointed to by directives will strengthen the quality of SRS [11].

2) RDQ related attributes

a) Readability Index

A category of attributes that measure how easily an adult can read and understand the requirements document [11].

Each of the above attributes had been associated with the Quality Properties under the Goal Properties as listed in Table 1.

Table 1
List of Quality Attributes

Aspects	Quality Properties	Quality Attributes
RSQ	Conciseness	▪ Implicit Sentences ▪ Optional Sentences ▪ Vague Sentences ▪ Weak Sentences
	Understandability	▪ Multiple Sentences
	Completeness	▪ Directives Sentences
RDQ	Understandability	▪ Readability Index

In Table 1, the SRS Quality Properties that had been identified were listed and mapped based on the previously defined Quality Attributes. The Quality Attributes that are feasible to be automatically calculated are mapped to the respective Quality Properties [10]. For example, conciseness quality property can be measured from the perspectives of implicit, optional, vague and weak sentences.

B. Existing Measurement Techniques

The existing measurement tools in the industry are Automated Requirements Measurement (ARM) tool [11] and Quality Analyzer of Requirements Specifications (QuARS) tool [10]. The ARM tool scans the designated file by the user as the text file that contains the requirements specification. The tool searches each text line of requirements specification for specific words and phrases [11]. The following words and phrases indicate the document's quality as a specification of requirements. In the end of the tool's process, it can generate a report contained imperative report and detailed weak phrase report.

In addition, QuARS is a prototype tool that automatically performs the quality evaluation of SRS by identifying the indicators and pointing out the sentences containing potential inaccuracies and ambiguities [10]. The document is used during review sessions where the output supports the reviewers in terms of detection of inaccuracies, ambiguities

and linguistic inconsistencies in the document.

III. RESEARCH METHODOLOGY

The generic methodology used in this work consists of five main phases, which are literature review, analyze rules, implementation, evaluation and documentation as visualized in Figure 1.

Figure 1: Research Methodology

A. *Literature Review*

The existing works on the related areas to measuring SRS quality had been investigated. The outcome of this is presented in Section II.

B. *Analyze Rule*

In order to determine the feasible rules, the Quality Attributes mentioned in both research papers were analyzed, compared and the similar rules were mapped to the same Quality Attributes as listed in Table 2. The table lists the rules to identify each of the Quality Attributes according to RSQ and RDQ [10] [11].

Table 2
Rules of Quality Attributes

Aspects	Quality Attributes	Rules
RSQ	Rule 1: Implicit Sentences Rule 2: Optional Sentences Rule 3: Vague Sentences Rule 4: Weak Sentences Rule 5: Multiple Sentences Rule 6: Directives Words	Refer to the corresponding terms in Table 3
RDQ	Rule 7: Readability Index	Flesch Reading Ease Readability. Calculate the total words, total sentences, and total syllables. Then include all the calculation using the following formula:- $206.835 - 1.015\left(\frac{total\ words}{total\ sentences}\right) - 84.6\left(\frac{total\ syllables}{total\ words}\right)$

For each of the rules, the respective terms which are applicable to the respective rules are listed in Table 3.

Table 3
Terms to Identify Error

Type	Words
Implicit Words	this, these, that, those, it, they, above, below, previous, next, following, last, and first.
Optional Words	can, eventually, if appropriate, if needed, may, optionally, possibly.
Vague Words	adequate, back, bad, clear, close, easy, efficient, far, fast, future, good, in front, low, near, new, old, past, recent, significant, slow, strong, today's, useful, weak, and well.
Weak Sentences	as a minimum, as applicable, as appropriate, be able to, be capable, not limited to, the capability of, the capability to, easy, effective, if practical, normal, provide for, to be determined, and timely.
Directives Words	figures, for example, table, note

Then, these rules (i.e. Rule 1-7) and terms (see Table 3) are designed to be implemented in the system.

C. *Implementation*

We developed a tool called SRS Quality Checker as a proof-of-concept of the rules, which measures the quality of SRS document based on RSQ and RDQ. Our system is developed by following the spiral model (see Figure 2). It is worth to point out that the spiral model used here is the software development methodology that we applied to guide us for the tool development only.

Each phase in the spiral model starts with a design goal and ends with the client (who may be internal) reviewing the progress [12]. The process begins at the center position. From there it moves clockwise in iterations. Each iteration of the spiral usually results in a deliverable.

In the first iteration, requirement specification is gathered for our system. This phase is needed as requirement specification is crucial for any development of system or product because it describes how a product will work.

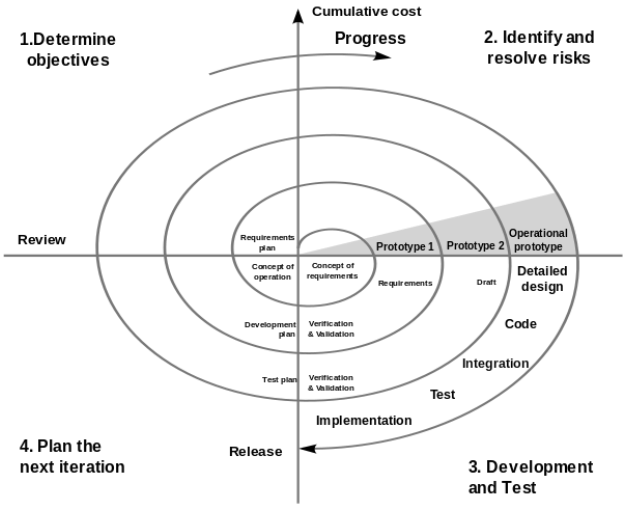

Figure 2: Spiral model

After specifying the requirements, the prototype is developed in the second iteration. Our prototype is developed according to its requirements. The algorithms for both the RDQ and RSQ properties are shown in Figure 3-6. Figure 3 and Figure 4 demonstrate how to measure RSQ, while Figure 5 explains how to measure RDQ. In order to measure RSQ, each requirement sentences in SRS is evaluated for each quality attributes that had been mentioned above (see Table 2).

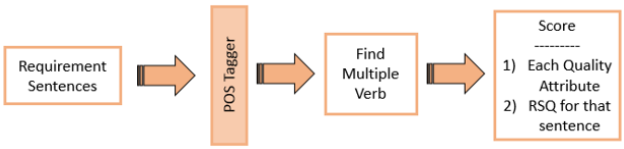

Figure 3: Algorithm for measuring multiple sentences

Figure 3 explains the algorithm that is for multiple sentences. See Rules 5. This rule implements the Natural Language Processing (NLP), which is Part-Of-Speech Tagger or called POS Tagger [11]. POS Tagger is a piece of software that reads the text in some language and assigns parts of speech to each word (and another token), such as noun, verb, adjective, etc [13].

Data: Each requirement sentence of the document, R
Result: Number of errors found in each requirement sentence
initialization;
while *not at end of the input document* **do**
  read POS tagger;
  if *word tagged equals VERB* **then**
    countVerb = countVerb - 1;
  end
  if *countVerb less than -1* **then**
    totalVerb = totalVerb + countVerb;
  end
end
totalError = totalError + totalVerb;
return totalError;

Algorithm 1: Pseudo code for Measuring Multiple Sentences

Figure 4: Algorithm for identifying multiple sentences

For this algorithm, firstly, each requirement sentence is read by the POS Tagger method then assigns parts of speech to each word. Then, the algorithm read one by one character of the requirement sentences to find the multiple verbs. If the requirement sentences have more than one verbs, it stated as having an error of multiple sentences. Due to space constraint, we only included the algorithm for this rule's implementation. See Figure 4.

Whilst, the algorithm in Figure 5 applies for all quality attributes except for multiple sentences. It compares every word in each requirement sentences with the database that contain the rules of quality attributes. This algorithm works, first of all, by splitting each requirement sentences into word.

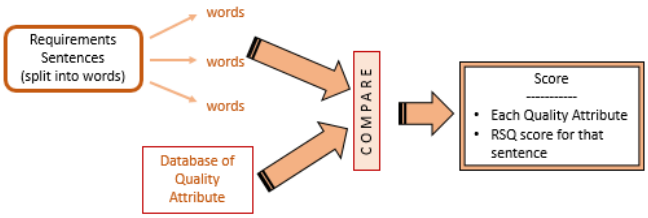

Figure 5: Algorithm for measuring the quality attribute except for multiple sentences

Then, it compares the split word with each rule in the database of quality attributes. If they match, the error word is displayed. For every error found, it is scored with a negative one and this is applied in both algorithms in Figure 5 and Figure 6. Each requirement sentence is free from error based on this work or rules.

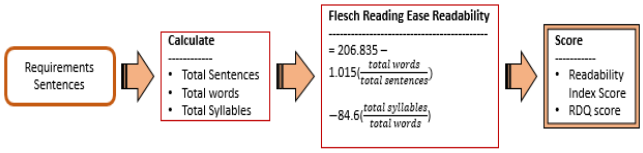

Figure 6: Algorithm for measuring readability index

To measure RDQ, Flesch Reading Ease is used as a rule to identify whether SRS document is readable by the user or not [11]. For this algorithm, total sentences, total words and total syllables of SRS document are calculated first. Then, they are included in the formula as in Figure 6. The score is calculated from the formula and used as an indicator to assess the ease of readability of a document.

For design, based on the algorithms, we used UML class diagram to visualize all the elements in the implementation. See Appendix for the class diagram of the SRS Quality Checker tool.

In the third iteration, the prototype is tested by using the real SRS document. It is worth to note here that a filtering process is required to ensure the input (i.e. SRS document) is in the correct format.

Whilst, in the fourth iteration, all of these changes from one iteration to another iteration are repeated until it leads to a system, which is functional. The snapshot of the tool can be viewed in Figure 7.

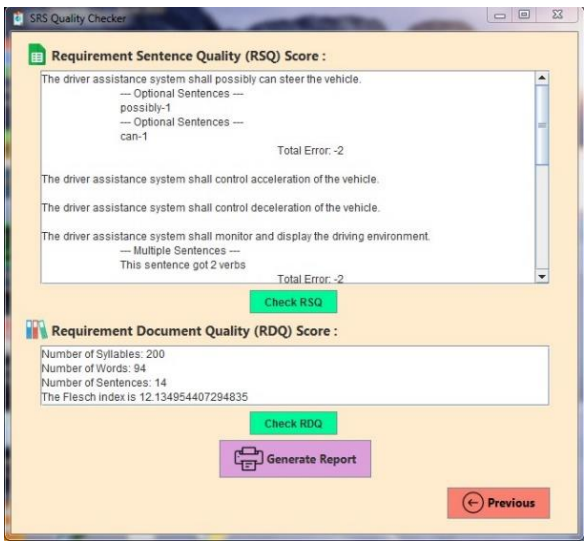

Figure 7: Snapshot of the outcome

Finally, the tool shall generate a summary report consisting of all the errors detected from the SRS document. See Figure 8. This allow the users to monitor the quality of their SRS document as well as updating the contents of the document to improve the SRS quality.

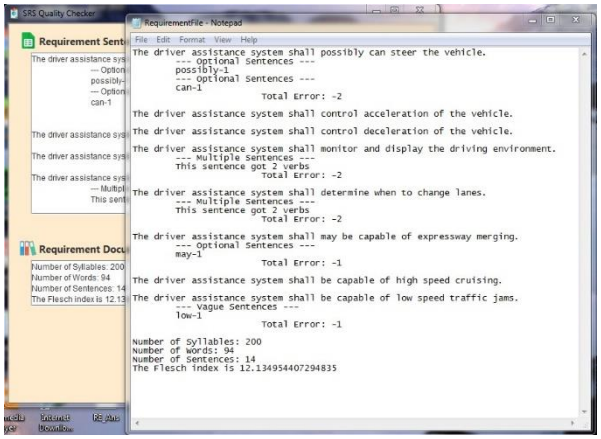

Figure 8: SRS quality report generator

D. Evaluation and Result

For every error found during measuring the RSQ properties, it is given the score of negative one and this is applied in both algorithms in Figure 3 and Figure 5. The errors are displayed in the tool according to their quality attributes. Each requirement sentence is free from error if it has a score of zero. In addition, the scores obtained after measuring RDQ can be interpreted as shown in Table 4.

Table 4
Score of Flesch Ease Readability Index [14]

Score	School Level	Notes
100.0–90.0	5th grade	Very easy to read.
90.0–80.0	6th grade	Easy to read.
80.0–70.0	7th grade	Fairly easy to read.
70.0–60.0	8th & 9th grade	Plain English.
60.0–50.0	10th to 12th grade	Fairly difficult to read.
50.0–30.0	College	Difficult to read.
30.0–0.0	College Graduate	Very difficult to read.

The Flesch Ease Readability Index [15] is based on the average number of syllables per word and the average number of words per sentence. Scores range from 0 to 100 with standard writing averaging 60 to 70. The higher the score, the greater the number of people who can readily understand the document [11].

IV. DISCUSSION

We are aware that this research is incomprehensive as there are Quality Attributes that are not feasible to be automated. In such cases, human reviewers are still required to make judgments. In addition, the feasibility of the approach is subjected to further evaluation by experts.

Nonetheless, our effort in conducting this research is at least can be useful for requirements engineers to measure their SRS quality. The implementation can benefit in measuring SRS quality for the applied attributes or even to be used in pre-review sessions.

V. CONCLUSION

This paper has analyzed and proposed a methodology to measure the SRS Quality according to Quality Attributes. We applied the methodology and rules to the SRS document and evaluated the results. The evaluation results proved that the methodology can be used as a framework and effort to measure the SRS quality. As a conclusion, by measuring the quality of SRS using rules, it could assist requirement engineers to improve their SRS quality.

APPENDIX

Appendix A - See Figure 9: Class Diagram for Implementation.

ACKNOWLEDGMENT

This work is sponsored by Research Initiative Grant (RIGS-16-344-0508). We would also like to thank the KICT faculty and IIUM for the opportunity to conduct this research.

REFERENCES

[1] P. Jalote, *An Integrated Approach to Software Engineering*, 3rd Edition. Narosa Publishing House, India, 2005.

[2] D. Pandey, U. Suman, and A. K. Ramani, "An Effective Requirement Engineering Process Model for Software Development and Requirements Management," in *Proceedings of the International Conference on Advances in Recent Technologies in Communication and Computing*, 2010, pp. 287–291.

[3] J. Siddiqi, "Requirement engineering: The emerging wisdom," *IEEE Softw.*, vol. 13, no. 2, pp. 15-19, Mar. 1996.

[4] D. Rubinstein, "Standish group report: There's less development chaos today," *Software Development Times*, vol. 1, 2007.

[5] A. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, A. Ta, and M. Theofanos, "Identifying and measuring quality in a software requirements specification," in *Proceedings of the First International Software Metrics Symposium*, 1993, pp. 141–152.

[6] "IEEE Recommended Practice for Software Requirements Specifications," *IEEE Std., 830-1998*, pp. 1–40, Oct. 1998.

[7] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specification*. 1st edition. NJ: Wiley, 2009.

[8] S. O. Mokhtar, R. Nordin, Z. A. Aziz and R. M. Rawi, "Issues and challenges of requirement review in the industry." *Indian Journal of Science and Technology*, vol. 10, no. 3, pp. 1-5, Jan. 2017.

[9] J. Krogstie, O. I. Lindland and G. Sindre, "Towards a deeper understanding of quality in requirements engineering," in *Proceedings of the International Conference on Advanced Information Systems Engineering*, 1995, pp. 82–95.

[10] F. Fabbrini, M. Fusani, S. Gnesi and G. Lami, "Quality evaluation of software requirement specifications," in *Proceedings of the Software and Internet Quality Week 2000 Conference*, 2000, pp. 1–18.

[11] Wilson, W. M., "Writing effective requirements specifications," *CrossTalk: The Journal of Defense Software Engineering*, pp. 16–19, 1999.

[12] V. Rastogi, "Software Development Life Cycle Models-Comparison, Consequences," *International Journal of Computer Science and Information Security*, vol. 6, no. 1, pp. 168–172, 2015.

[13] S. Mall and U. C. Jaiswal, "Evaluation for POS tagger, chunk and resolving issues in word sense disambiguate in machine translation for Hindi to English languages," in *Proceedings of the 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2016, pp. 14–18.

[14] R. Flesch, "How to Write Plain English". University of Canterbury. Available at http://www.mang.canterbury.ac.nz/writing_guide/writing/flesch.shtml. [Retrieved 5 February 2016].

[15] J. P. Kincaid, J. Fishburne, R. L. Rogers and B. S. Chissom, *Derivation of New Readability Formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel*. University of Central Florida, Feb. 1975.

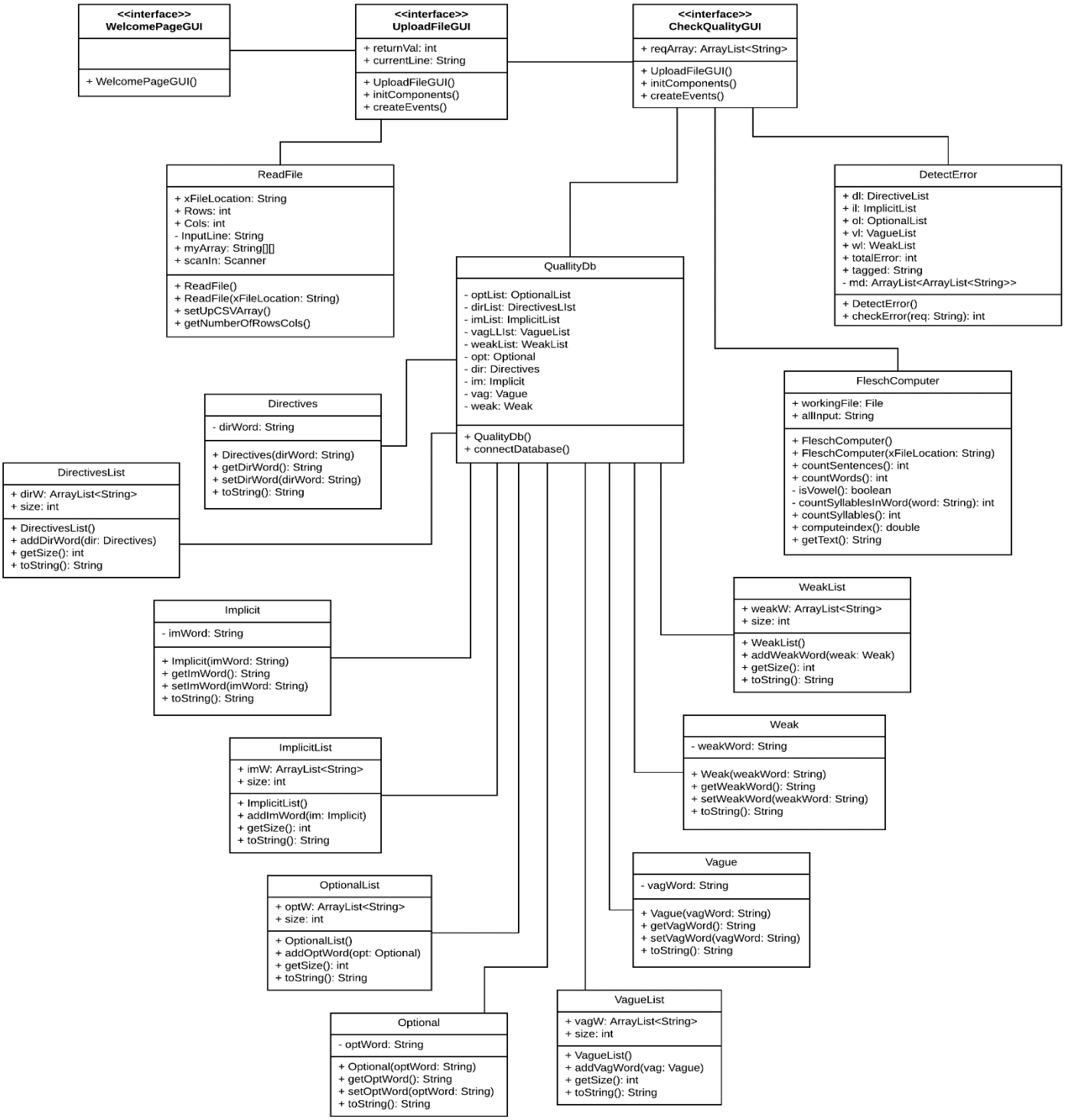

Figure 9: Class Diagram for the Implementation